Notice of Allowability	Application No.	Applicant(s)	
	10/675,016	GALVIN ET AL.	
	Examiner	Art Unit	
	Satish S. Rampuria	2191	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address--

All claims being allowable, PROSECUTION ON THE MERITS IS (OR REMAINS) CLOSED in this application. If not included herewith (or previously mailed), a Notice of Allowance (PTOL-85) or other appropriate communication will be mailed in due course. **THIS NOTICE OF ALLOWABILITY IS NOT A GRANT OF PATENT RIGHTS.** This application is subject to withdrawal from issue at the initiative of the Office or upon petition by the applicant. See 37 CFR 1.313 and MPEP 1308.

1. ☒ This communication is responsive to *05/23/2007*.

2. ☒ The allowed claim(s) is/are *1.4-15,17-32 (Renumbered as 1-29)*.

3. ☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).
   a) ☐ All b) ☐ Some* c) ☐ None of the:
      1. ☐ Certified copies of the priority documents have been received.
      2. ☐ Certified copies of the priority documents have been received in Application No. _____ .
      3. ☐ Copies of the certified copies of the priority documents have been received in this national stage application from the International Bureau (PCT Rule 17.2(a)).

   * Certified copies not received: _____.

Applicant has THREE MONTHS FROM THE "MAILING DATE" of this communication to file a reply complying with the requirements noted below. Failure to timely comply will result in ABANDONMENT of this application.
THIS THREE-MONTH PERIOD IS NOT EXTENDABLE.

4. ☐ A SUBSTITUTE OATH OR DECLARATION must be submitted. Note the attached EXAMINER'S AMENDMENT or NOTICE OF INFORMAL PATENT APPLICATION (PTO-152) which gives reason(s) why the oath or declaration is deficient.

5. ☐ CORRECTED DRAWINGS (as "replacement sheets") must be submitted.
   (a) ☐ including changes required by the Notice of Draftsperson's Patent Drawing Review (PTO-948) attached
      1) ☐ hereto or 2) ☐ to Paper No./Mail Date _____.
   (b) ☐ including changes required by the attached Examiner's Amendment / Comment or in the Office action of Paper No./Mail Date _____.

   Identifying indicia such as the application number (see 37 CFR 1.84(c)) should be written on the drawings in the front (not the back) of each sheet. Replacement sheet(s) should be labeled as such in the header according to 37 CFR 1.121(d).

6. ☐ DEPOSIT OF and/or INFORMATION about the deposit of BIOLOGICAL MATERIAL must be submitted. Note the attached Examiner's comment regarding REQUIREMENT FOR THE DEPOSIT OF BIOLOGICAL MATERIAL.

Attachment(s)

1. ☒ Notice of References Cited (PTO-892)
2. ☐ Notice of Draftperson's Patent Drawing Review (PTO-948)
3. ☐ Information Disclosure Statements (PTO/SB/08), Paper No./Mail Date _____
4. ☐ Examiner's Comment Regarding Requirement for Deposit of Biological Material
5. ☐ Notice of Informal Patent Application
6. ☒ Interview Summary (PTO-413), Paper No./Mail Date *6/14/07* .
7. ☒ Examiner's Amendment/Comment
8. ☐ Examiner's Statement of Reasons for Allowance
9. ☐ Other _____.

DETAILED ACTION

This action is in response to the amendment filed on 05/23/2007.

Claims 1, 4-15,17-32 are allowed.

Claims 3 and 16 are cancelled by the Applicant.

Claim 2 is cancelled with this Examiner's amendment.

Claims 1, 8, 15, 22, 29 and 30-32 are amended by the Applicant.

Examiner's Amendment

An examiner's amendment to the record appears below. Should the changes and/or additions be unacceptable to applicant, an amendment may be filed as provided by 37 CFR 1.312. To ensure consideration of such an amendment, it MUST be submitted no later than the payment of the issue fee.

Authorization for this examiner's amendment was given in a telephone interview with LUTZ, CHRISTOPHER [Reg. No. 44,883] on June 14, 2007.

In the Abstract:

Please replace the abstract with the following.

Developing web interfaces to existing base applications encounter GUI screens involves translation from the native language into HTML (Hypertext Markup Language) to support operation from a web page in accessible by HTTP (Hypertext Transfer Protocol), as is common to Internet (web based) applications. Accordingly, such "webification" of a conventional application involves manual ~~porting the base application by manually hand-coding the references to a web based GUI~~. An object translation mechanism allows a web server to invoke the base application via a web GUI by using a server runtime engine in the web server to generate transportable objects, corresponding to application objects, for transmission to a client runtime engine (Browser). The web server further receives return transportable objects and generates executable objects indicative of user input and commands. The executable objects, ~~indicative of user input,~~ map the user inputs and objects to associated objects and external references in the web server application corresponding to the base application, and perform the manipulations and operations corresponding to the user input.

In the Specification:

Please replace the paragraph on page 2, lines 27-31 to page 3 lines 1-6 as follows.

In a conventional information processing system, a developer (typically a programmer or engineer) models the system as a set of deployment objects, typically software objects based on an object oriented design for deployment using an object-oriented language such as ~~Java~~ Java™ or C++ ("Java" is a registered Trademark of Sun Microsystems, Inc., of Santa Clara, CA, for its web based programming language, as is known in the art). Such objects are executable on a conventional interpreter such as a runtime engine compatible with the language and responsive to the objects. A developer models, develops, and deploys a set of such conventional objects as an application. Typically, a set of objects defining a particular conventional application employ a graphical user interface (GUI) operable from a user console for displaying and receiving information from a user or operator (user). The GUI provides the HMI interface to the conventional application, which ultimately performs a particular task, such as managing a data source or providing a service on behalf of the user.

Please replace the paragraph on page 5, lines 15-22 as follows.

One such application which can assist in converting software objects, such as ~~Java~~ Java™ objects, into transmission units operable for transmission via a public access network, is the Nexaweb Smart Client software platform, marketed commercially by Nexaweb Technologies, Inc., of Cambridge, Massachusetts. The Nexaweb product purports to deliver a Client/Server experience over the Web. However, the Nexaweb platform does not appear to provide a high level object library, but rather retains the lower level metalanguage approach to manually map XML objects, therefore tending to mitigate automated or seamless integration with external servers and data repositories.

Please replace the paragraphs on page 6, lines 1-19 as follows.

In a particular configuration, the executable objects are ~~Java~~ Java™ objects and the server runtime engine is operable to generate transportable objects in XML, and transmit the XML documents to the remote client runtime engine for supporting a web based (Internet) GUI. The remote client employs a browser with the client runtime engine to support the web-based GUI. The transportable objects are XML using the XUL syntax (schema), as is known to those of skill in the art, to communicate with the web server, collectively forming the web application. The "instructions" in the transportable objects are therefore defined in terms of the metalanguage, in this case XUL as opposed to HTML.

The system, therefore, allows the application developer (programmer) to design and program the GUI strictly in ~~Java~~Java™ using Java classes in order to express the view of the base application over the web. Developers need not think in terms of a metalanguage; they think in terms of high level objects (Java GUI classes). The result is a substantial departure from conventional methodologies which require applications to map their business objects to a metalanguage by hand. A programmer or developer creates high level GUI objects, which run as executable objects on the web server. Metalanguage encoded representations of the objects are transmitted to a remote client as transportable objects. From the GUI screens, responses and client requests are received and mapped back into the high level GUI objects at the server.

Please replace the paragraph on page 8, lines 22-29 as follows.

Translating the objects for deployment and execution on the web server includes determining, for each of the translated application objects, overloaded methods corresponding to GUI display elements. An overload parser

in the object classifier parses the GUI objects and the corresponding display elements, and resolves style inconsistencies in the GUI display produced by the client runtime engine. The base application may employ overloaded methods, a practice common with ~~Java~~Java™ and other object-oriented implementation languages. The overload parser resolves undetermined references to methods resulting from such overload practices.

Please replace the paragraph on page 9, lines 27-31 to page 10, lines 1-13 as follows.

The invention as disclosed above is described as implemented on a computer having a processor, memory, and interface operable for performing the steps and methods for deploying a remote deployment system as disclosed herein. Other embodiments of the invention include a computerized device such as a computer system, central processing unit, microprocessor, controller, electronic circuit, application-specific integrated circuit, or other hardware device configured to process all of the method operations disclosed herein as embodiments of the invention. In such embodiments, the computerized device includes an interface (e.g., for receiving data or more segments of code of a program), a memory (e.g., any type of computer readable medium), a processor and an interconnection mechanism connecting the interface, the processor and the memory. In such embodiments, the memory system is encoded with an application having components that when performed on the processor, produces a process or processes that causes the computerized device to perform any and/or all of the method embodiments, steps and operations explained herein as embodiments of the invention to allow execution of instructions in a computer program such as a ~~Java~~Java™ application. In other words, a computer, processor or other electronic device that is programmed to operate embodiments of the invention as explained herein is itself considered an embodiment of the invention.

Please replace the paragraph on page 11, lines 20-31 to page 12, lines 1-7 as follows.

In a particular exemplary configuration, the executable objects are ~~Java~~Java™ objects and the server runtime engine is operable to generate transportable objects in XML, transmit the XML documents to the remote client runtime engine for supporting a web based (Internet) GUI. The remote client employs the browser with the client runtime engine to support the web-based GUI. The transportable objects are XML using the XUL syntax (schema), as is known to those of skill in the art, to communicate with the web server, collectively forming the web application. The "instructions" in the transportable objects are therefore defined in terms of in a metalanguage, in this case XUL as opposed to HTML.

The system, therefore, allows the application developer (programmer) to design and program the GUI in ~~Java~~Java™ in order to express the view of the base application over the web. Developers need not think in terms of a metalanguage; rather they think in terms of high level objects (Java GUI classes). The result is a substantial departure from conventional methodologies which require applications to map their business objects to a metalanguage by hand. A programmer or developer creates high level GUI objects, which run as executable objects on the server. Meta-language encoded representations of the objects are transmitted to a remote client as transportable objects. From the GUI screens, responses and client requests are received and mapped back into the high level GUI objects at the server.

Please replace the paragraph on page 17, lines 25-31 to page 18, lines 1-8 as follows.

At steps 215-223, the base application 16 has additional operations and functions having different behavior at the web server 32 than at the base application server 19. Accordingly, the object translator 26 selectively modifies the remote application objects 50 for appropriate operation in the server runtime engine 46. At step 216, translating includes determining overloaded methods corresponding to GUI display elements. Certain implementation languages for implementing the objects, such as ~~Java~~Java™ and c++, employ overloaded methods, or functions, which allows invocation from among multiple similarly named methods to match the type of data

passed to the operation in the object 14. An overload parser 76 in the object classifier 64 determines, for each of the translated application objects 50, inconsistent or improper overload references. Such an improper overload reference may include, for example, an overloaded method which receives the incorrect object having data items, or attributes, which are better handled by another method. Display anomalies or inconsistencies may result from such inappropriate overload usage, such as invoking a display element 37' in an alternative style or color by the client runtime engine 38.

In the claims:

Please amend claims 1, 30 and 31 as follows.

1. (Currently Amended) A method of modeling, building and implementing a software application on a remote deployment system corresponding to a base application comprising:

identifying a set of objects in the base application for inclusion in the remote deployment and operable by an alternate control path;

translating, via an object translator, the identified set of objects into a set of remote application objects parallel to the objects in the base application, the identified set of objects defining a graphical user interface operable to interact with a user;

wherein translating comprises generating, via a label mapper in the object translator, a corresponding remote application object for each identified object in the base application, the generated remote application object operable for execution in the remote deployment;

deploying the translated remote application objects on a remote server; and

generating, from at least a subset of the translated remote application objects, executable objects executable by a server runtime engine at the remote server, the server runtime engine operable to generate transportable objects corresponding to the generated executable objects, the transportable objects further operable to generate, via the alternate control path, GUI executable objects on a remote client runtime engine, the remote client runtime engine responsive to the transportable objects to generate the corresponding GUI executable objects, the set of objects in the base application further comprising GUI objects and processing objects, translating further comprising:

determining, via an object classifier in the object translator, if the object is a GUI object or a processing object; and

if the object is a GUI object, generating a reference to the server runtime engine.

30. (Currently Amended) A computer program product having a computer readable storage medium operable to store computer program logic embodied in an encoded set of processor based instructions defined as computer program code encoded thereon and executable by a processor responsive to the instructions for modeling, building and

implementing a software application on a remote deployment system corresponding to a base application comprising:

computer program code for identifying a set of objects in the base application for inclusion in the remote deployment and operable by an alternate control path;

identifying, from the set of objects in the base application, GUI objects and processing objects, the GUI objects responsible for producing GUI display elements;

computer program code for translating, via an object translator, the identified set of objects into a set of remote application objects parallel to the objects in the base application, the identified set of objects defining a graphical user interface operable to interact with a user;

wherein translating comprises generating, via a label mapper in the object translator, a corresponding remote application object for each identified object in the base application, the generated remote application object operable for execution in the remote deployment;

computer program code for deploying the translated remote application objects on a remote server; and

computer program code for generating, from at least a subset of the translated remote application objects, executable objects executable by a server runtime engine at the remote server, the server runtime engine operable to generate transportable objects corresponding to the generated executable objects, the transportable objects further operable to generate, via the alternate control path, GUI executable objects on a remote client runtime engine, the remote client runtime engine responsive to the transportable objects to generate the corresponding GUI executable objects, translating further comprising:

identifying, via an association manager in the object translator, associations between the remote application objects and the GUI display elements; and

storing, in an associated object table, the identified associations.

31. (Currently Amended) An encoded set or processor based instructions on a computer readable storage medium defined as program code executable by a processor responsive to the instructions computer data signal for modeling, building and implementing a software application on a remote deployment system corresponding to a base application comprising:

program code for identifying a set of objects in the base application for inclusion in the remote deployment and operable by an alternate control path;

program code for translating, via an object translator, the identified set of objects into a set of remote application objects parallel to the objects in the base application, the identified set of objects defining a graphical user interface operable to interact with a user;

wherein translating comprises generating, via a label mapper in the object translator, a corresponding remote application object for each identified object in the base application, the generated remote application object operable for execution in the remote deployment;

program code for deploying the translated remote application objects on a remote server; and

program code for generating, from at least a subset of the translated remote application objects, executable objects executable by a server runtime engine at the remote server, the server runtime engine operable to generate transportable objects corresponding to the generated executable objects, the transportable objects further operable to generate, via the alternate control path, GUI executable objects on a remote client runtime engine, the remote client runtime engine responsive to the transportable objects to generate the corresponding GUI executable objects.

--END--

Conclusion

The prior art made of record and not relied upon is considered pertinent to applicant's disclosure.

Any inquiry concerning this communication or earlier communications from the examiner should be directed to Satish S. Rampuria whose telephone number is (571) 272-3732. The examiner can normally be reached on 8:30 am to 5:00 pm.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Wei Y. Zhen can be reached on (571) 272-3708. Any inquiry of a general nature or relating to the status of this application should be directed to the TC 2100 Group receptionist: 571-272-2100.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

Satish S. Rampuria
Patent Examiner/Software Engineer
Art Unit 2191

WEI ZHEN
SUPERVISORY PATENT EXAMINER